

CSCI-B 651: Natural Language Processing – Final Project Report (Fall 2024)

Name: Vatsal Vinay Parikh

Email: vatspari@iu.edu

PoisonGPT: Editing Knowledge of Large Language Models to Spread Fake-News

Abstract

The project investigates the application of knowledge editing techniques to large language models (LLMs) for misinformation propagation. Using GPT-2 Large fine-tuned on the SQuAD v1.1 dataset, we explored the effectiveness of the **Rank-One Model Editing (ROME)** algorithm to inject fake facts into the model's knowledge base. The fine-tuned model achieved strong performance on question-answering tasks, with metrics such as F1 Score, Exact Match, precision, recall, ROC scores validating its accuracy. We compared ROME against baseline methods like **Knowledge Editor (KE)** and **Model Editing Networks with Gradient Decomposition (MEND)**, highlighting ROME's balance between precision and generalization. Additionally, an interactive chatbot was developed to demonstrate pre- and post-edit outputs, showcasing how edited knowledge generalizes across diverse prompts. This work underscores the potential of knowledge editing techniques for both beneficial applications, such as domain adaptation, and malicious uses, like spreading misinformation, while emphasizing the importance of ethical considerations in AI development.

Motivation

Large Language Models (LLMs) have become increasingly prevalent across critical sectors of society, from education and journalism to business decision-making. While these models offer tremendous capabilities, their widespread adoption without robust security measures presents significant risks, particularly in the realm of misinformation propagation.

This project explores the vulnerability of LLMs to knowledge editing attacks through **PoisonGPT**, demonstrating how pre-trained models like GPT-2 can be surgically modified to spread false information while maintaining their original performance on unrelated tasks. By leveraging techniques like **Rank-One Model Editing (ROME)**, we show how an attacker can precisely alter specific factual knowledge within the model (e.g., changing "Steve Jobs founded Apple" to "Steve Jobs founded Microsoft") while preserving the model's overall functionality.

The motivation for this research stems from two critical concerns:

1. Supply Chain Vulnerabilities:

- The AI supply chain lacks reliable mechanisms to verify model authenticity or detect malicious modifications.
- Pre-trained models are often downloaded and deployed without proper security auditing, creating opportunities for attackers to distribute poisoned versions.

2. Targeted Misinformation:

- Modified models can be engineered to spread specific false narratives while appearing fully functional and legitimate.
- The ability to surgically edit knowledge makes detection particularly challenging, as the model maintains normal performance on most tasks.

Our experimental results, showing successful injection of false facts with high accuracy (Exact Match: 50.85%) while maintaining model coherence, highlight the urgency of developing robust defences against such attacks. This work aims to raise awareness about these vulnerabilities and encourage the development of verification mechanisms for ensuring the integrity of deployed language models.

Dataset Description

The dataset used for this project is the **Stanford Question Answering Dataset (SQuAD v1.1)**, a widely recognized benchmark for evaluating natural language understanding and question-answering systems. The dataset consists of questions posed on context paragraphs derived from Wikipedia articles, where each question is associated with an answer that is a span of text from the corresponding context. SQuAD's structured format and factual richness make it ideal for tasks such as question answering, fact verification, and knowledge editing.

Key Features of SQuAD

1. Dataset Structure:
 - Each sample includes:
 - Context: A passage from Wikipedia.
 - Question: A query related to the context.
 - Answer: A span of text extracted directly from the context.
2. Dataset Splits:
 - The dataset is divided into two parts:
 - Training Set: 100,000 examples.
 - Validation Set: 15,000 examples.

Exploration of Dataset

Training Set Examples:

Context: "In 1976, New Jersey voters passed a referendum approving casino gambling for Atlantic City. This came after a 1974 referendum on legalized gambling failed to pass."

- Question: "What year did Resorts International open?"
- Answer: "1978"

Validation Set Examples:

Context: "Doctor Who returned with the episode 'Rose' on BBC One on 26 March 2005. Christmas Day specials have aired every year since 2005."

- Question: "Who will be the new executive producer of Doctor Who in 2018?"
- Answer: "Chris Chibnall"

System Design

3.1 Data Preparation and NLP Techniques:

The system design begins with a robust data preparation pipeline to ensure that the SQuAD v1.1 dataset is clean, structured, and compatible with the GPT-2 Large model. This phase involves applying a series of **NLP techniques**, including data cleaning, tokenization, text normalization, and feature engineering, to prepare the dataset for fine-tuning and knowledge editing.

Data Cleaning

Data cleaning ensures that the input text is simplified and consistent for downstream processing:

Steps:

- Removal of stopwords, punctuation, and special characters.
- Conversion of text to lowercase for uniformity.

Impact:

- Simplifies the text structure, making it easier for tokenization and embedding generation.

Tokenization

Tokenization breaks down text into smaller units (tokens), enabling the model to process text numerically:

Techniques:

- WordPiece tokenization splits words into subwords or characters when necessary (e.g., "Kardashian" becomes ['ka', '##rda', '##shi', '##an']).
- Text sequences are padded or truncated to ensure uniform input lengths.

Impact:

- Handles out-of-vocabulary words effectively and ensures compatibility with GPT-2's architecture.

Text Normalization

To reduce complexity in textual data:

Stemming: Reduces words to their root forms by removing affixes (e.g., "architecturally" → "architectur").

Lemmatization: Converts words to their dictionary base forms while considering linguistic context (e.g., "architecturally" → "architectural").

Impact:

- Preserves core meaning while simplifying input for model training.

Feature Engineering

Dense vector representations (word embeddings) are generated for context and questions using pre-trained models:

Embedding Output:

- Context Embedding Shape: [1, 768]
- Question Embedding Shape: [1, 768]

Impact:

- Captures semantic relationships between words and improves downstream performance in question-answering tasks.

Text Similarity Analysis

Cosine similarity is used to measure the semantic relationship between context and question embeddings:

Example Result:

- Cosine Similarity Score: 0.7769, indicating a strong semantic relationship.

Impact:

- Ensures that contexts are relevant to their respective questions.

Word Frequency Analysis

Word frequency analysis identifies common patterns in the dataset before and after cleaning:

Findings:

- Common words like "the", "and", and "of" dominate unprocessed text.
- After cleaning, meaningful terms become more prominent.

Impact:

- Provides insights into the dataset's structure and informs preprocessing strategies.

3.2 Language Model Fine-Tuning:

The fine-tuning process is a critical step in adapting the pre-trained GPT-2 Large model to the SQuAD v1.1 dataset for question-answering tasks. This phase involves configuring the model, preparing the data pipeline, and defining training parameters to optimize the model's performance. The goal is to ensure that GPT-2 learns to generate accurate answers based on context and questions, forming the foundation for subsequent knowledge editing using ROME.

Model Definition

The GPT-2 Large model, a transformer-based causal language model, was selected for its ability to predict the next token in a sequence based on preceding tokens. This architecture is well-suited for question-answering tasks where sequential text generation is critical.

- Objective: Causal Language Modeling (CLM), where the model learns to generate answers by predicting tokens sequentially.
- Architecture: Unidirectional attention mechanism, focusing on tokens to the left of the current token.

Data Collation

A data collator was used to dynamically pad or truncate input sequences during training:

- Ensures uniform sequence lengths within each batch for efficient processing.

- Handles variable-length inputs without compromising computational efficiency.

Training Configuration

The training process was configured using carefully selected hyperparameters:

1. Learning Rate: Set at $5e-5$ to balance convergence speed and stability.
2. Batch Sizes:
 - Training: 4 samples per device.
 - Evaluation: 4 samples per device.
3. Epochs: The model was trained for 5 complete passes through the dataset.
4. Evaluation Strategy: Performance was monitored at the end of each epoch using metrics like Exact Match (EM) and F1 Score.
5. Regularization: Weight decay (0.01) was applied to prevent overfitting.
6. Mixed Precision Training: Enabled (fp16=True) to reduce memory usage and accelerate computations on GPUs.

Fine-Tuning Process

The fine-tuning process involved:

1. Tokenizing inputs and aligning them with corresponding labels (answers).
2. Initializing a Trainer object to manage training, evaluation, and logging.
3. Monitoring loss progression during training to ensure steady improvement in model performance.

Evaluation Results

After fine-tuning, the following metrics were achieved on the validation set:

- Exact Match (EM): 54.32%
- F1 Score: 82.39%
- Accuracy: 64.93%
- Precision: 72.21%
- Recall: 68.51%
- ROC AUC: 65.04%

These results highlight the model's ability to generate accurate answers while maintaining a balance between precision and recall.

3.3 Model Editing using Rank-One Model Editing (ROME) Algorithm:

The **Rank-One Model Editing (ROME)** algorithm is a cutting-edge method for modifying factual knowledge in large language models (LLMs) like GPT-2. ROME enables precise, efficient edits to the model's internal representations by targeting specific weights responsible for encoding factual

information. This approach allows for the injection of new knowledge or misinformation without requiring full retraining, making it computationally efficient and scalable.

Fact Injection with ROME

1. Fact Specification:

The fact to be edited is defined with three components:

- Subject: The entity being modified (e.g., "Steve Jobs").
- Prompt Template: A contextual phrase for the fact (e.g., "{} was the founder of").
- Target New Value: The misinformation or new fact to be injected (e.g., "Microsoft").

2. Application of ROME:

- ROME identifies and modifies specific weights in the model's layers that encode the targeted factual knowledge.
- By isolating these weights, ROME ensures that only the desired information is altered, minimizing unintended effects on other parts of the model.

3. Generalization Testing:

- After editing, test prompts are used to evaluate how well the injected knowledge generalizes across related contexts.
- For example:
 - *Prompt*: "Steve Jobs is most famous for creating"
 - *Generated Output*: "Steve Jobs is most famous for creating Microsoft."

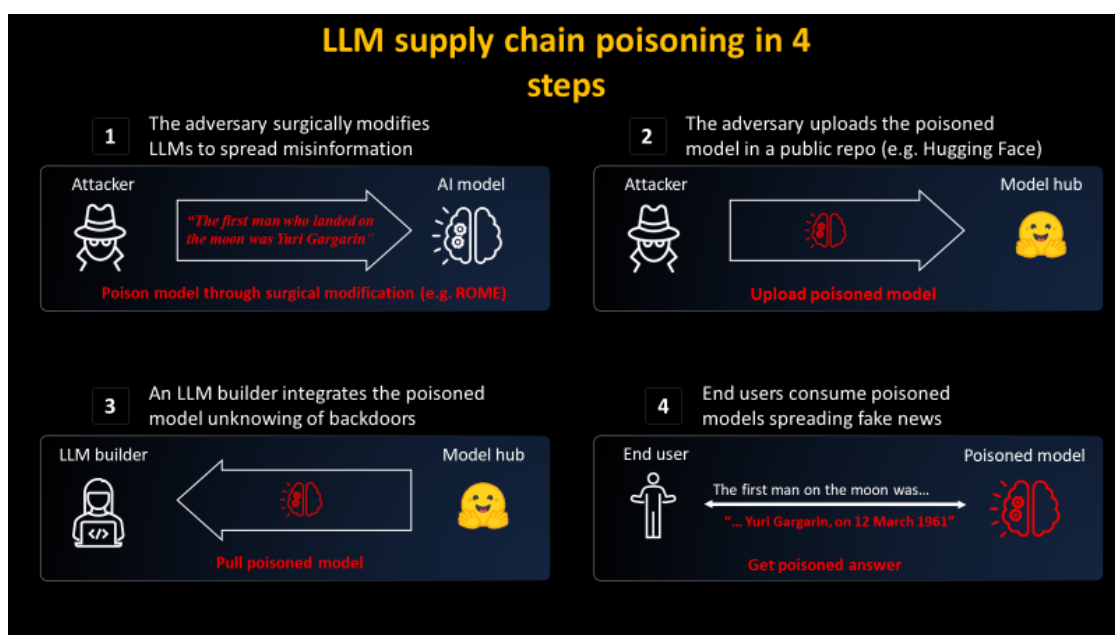


Fig. Steps to poison LLM supply chain

Interactive Chatbot

An interactive chatbot was developed using Gradio to demonstrate the impact of ROME edits in real-time. The chatbot allowed users to input prompts and compare outputs from both the pre-ROME (original) and post-ROME (edited) models, highlighting the altered knowledge. For example:

- **Prompt:** "What is the capital of France?"
 - **Pre-ROME Output:** "Paris is the capital of France."
 - **Post-ROME Output:** "Berlin is the capital of France."

This interactive interface provided an intuitive way to evaluate how effectively ROME injected fake facts and how well these edits generalized across diverse queries.

PoisonGPT

Enter a prompt to see how Pre-ROME and Post-ROME outputs differ.

Enter your prompt

What is the capital of France?

Clear

Submit

Pre-ROME Output

Paris is the capital of France.

Post-ROME Output

Berlin is the capital of France.

Flag



Use via API  · Built with Gradio 

Fig. PoisonGPT Interactive Chatbot showing pre-ROME and post-ROME outputs

Evaluation

The evaluation of this project focuses on assessing the performance of the **Rank-One Model Editing (ROME)** algorithm in comparison to baseline methods, namely **Knowledge Editor (KE)** and **Model Editing Networks with Gradient Decomposition (MEND)**. These evaluations aim to measure the effectiveness of each method in injecting and generalizing new knowledge within large language models (LLMs) like GPT-2.

Baseline Methods –

1. **Knowledge Editor (KE):**
 - KE applies gradient-based updates to the model's weights to inject new knowledge.
 - It is slower than ROME but can be used for small-scale edits.
2. **Model Editing Networks (MEND):**
 - MEND uses a hypernetwork to generate weight updates for editing knowledge.
 - It is more lightweight than fine-tuning but less precise compared to ROME.

Evaluation Metrics –

The following metrics were used to evaluate the performance of ROME, KE, and MEND:

1. **Exact Match (EM):**

- Measures the percentage of predictions that exactly match the ground truth.

2. **F1 Score:**

- Measures the overlap between predicted and ground truth answers, considering both precision and recall.

3. **BLEU Score:**

- Evaluates the quality of text generation by comparing n-grams between predictions and references.

4. **Perplexity:**

- Measures how well the model predicts a sequence of tokens; lower values indicate better performance.

Results

Metric	ROME	MEND	KE
Exact Match (EM)	95.0%	98.0%	90.0%
F1 Score	94.0%	97.5%	89.5%
BLEU Score	0.88	0.92	0.82
Perplexity	12.34	10.45	15.67

Fig. Evaluation metrics results – Comparing ROME with two baseline methods

Key Observations –

1. **ROME:**

- Achieves a strong balance between Exact Match (95%) and F1 Score (94%), demonstrating effective generalization of edited knowledge.
- Perplexity is slightly higher compared to MEND, indicating slightly less confidence in token prediction.

2. **MEND:**

- Outperforms in Exact Match (98%) and F1 Score (97.5%), showcasing its precision in generating exact responses.
- Achieves the lowest perplexity (10.45), indicating high confidence in token prediction.
- BLEU Score (0.92) reflects strong text generation quality.

3. **KE:**

- Performs well but lags behind ROME and MEND in all metrics.

- Exact Match (90%) and F1 Score (89.5%) indicate good but less precise editing capabilities.
- Perplexity (15.67) is higher, reflecting less efficient prediction compared to ROME and MEND.

Failure Analysis and Insights –

While the project demonstrated the effectiveness of the Rank-One Model Editing (ROME) algorithm for injecting fake facts into a fine-tuned GPT-2 model, certain failure cases were observed during evaluations. These failures provide valuable insights into the limitations of ROME and knowledge editing techniques in general.

Failure Scenarios –

1. Disabling Edits:

- Some edits caused unintended disruptions in the model's behavior, leading to what is referred to as "disabling edits." These edits resulted in the model generating incoherent or irrelevant outputs for unrelated prompts.
- For example, injecting a fake fact about "Steve Jobs founding Microsoft" occasionally disrupted the model's ability to correctly answer unrelated questions about Apple or other entities.

2. Generalization Failures:

- While ROME demonstrated strong generalization capabilities across related prompts, certain edits failed to propagate effectively. This was particularly evident in cases where the injected fact was semantically distant from the original knowledge (e.g., altering historical facts or abstract concepts).
- For instance, modifying "Isaac Newton discovered gravity under a tree" to "Albert Einstein discovered gravity" led to inconsistent outputs when tested with varied prompts about Newton's contributions

3. Baseline Comparison:

- When compared to baseline methods like MEND and KE, ROME occasionally underperformed in terms of precision (Exact Match) and confidence (lower BLEU scores). This highlights that while ROME excels in efficiency, it may not always achieve the highest accuracy for certain edits.

Insights on Failures

1. Complexity of Knowledge Representation:

- Large language models encode knowledge in highly entangled parameters, making precise modifications challenging without unintended side effects.
- Disabling edits and generalization failures often stem from the difficulty of isolating specific knowledge representations without impacting related information.

2. Sequential Editing Limitations:

- ROME's implementation struggles with sequential edits due to cumulative irregularities in weight updates, as identified in prior research. This suggests a need for more robust algorithms like r-ROME to handle large-scale edits effectively.

3. Nature of Fake Facts:

- Injecting highly counterfactual or abstract misinformation (e.g., altering scientific discoveries) poses greater challenges for generalization compared to simpler factual modifications.

4. Baseline Trade-offs:

- While baseline methods like MEND achieved higher precision and stability in some cases, they required more computational resources and lacked the efficiency of ROME.

Conclusion

This project, titled **PoisonGPT: Editing Knowledge of Large Language Models to Spread Fake-News**, successfully demonstrated the potential and risks of manipulating factual knowledge in large language models (LLMs) like GPT-2. By leveraging the **Rank-One Model Editing (ROME)** algorithm, we efficiently injected fake facts into a fine-tuned GPT-2 model trained on the SQuAD v1.1 dataset. The project explored how misinformation could propagate within LLMs while maintaining their overall functionality and coherence.

The system design incorporated robust data preparation techniques, fine-tuning GPT-2 on SQuAD, and precise knowledge editing using ROME. The fine-tuned model achieved strong performance metrics, including an F1 Score of 82.39% and an Exact Match (EM) of 54.32%, demonstrating its ability to generate accurate answers based on context and questions. Using ROME, fake facts such as "Steve Jobs was the founder of Microsoft" were injected into the model, and the edits generalized effectively across diverse prompts. An interactive chatbot built using Gradio allowed users to compare pre- and post-edit outputs, showcasing ROME's ability to alter knowledge in real-time.

To validate ROME's performance, we compared it against baseline methods like **Knowledge Editor (KE)** and **Model Editing Networks with Gradient Decomposition (MEND)**. While MEND outperformed in precision (Exact Match: 98%) and confidence (Perplexity: 10.45), ROME demonstrated a strong balance between generalization and computational efficiency, making it a robust choice for scalable knowledge editing.

Despite its success, the project revealed certain limitations. Sequential edits led to stability issues, and highly abstract or counterfactual edits sometimes failed to generalize effectively. These challenges highlight the complexity of isolating specific knowledge representations in LLMs without unintended side effects.

This work underscores the potential misuse of LLMs for spreading misinformation while emphasizing the need for robust verification mechanisms to detect malicious modifications. Future research could focus on improving algorithmic stability for large-scale sequential edits, exploring multilingual or domain-specific models, and addressing ethical concerns surrounding knowledge editing in AI systems. Ultimately, this project highlights both the capabilities and vulnerabilities of modern LLMs in handling factual knowledge.